Abstract

The aim of my project is to produce a documentation on how to build and maintain a stable and secure Linux Operating System. There are several different distributions I could have chosen for my project, but in my opinion, the best way in achieving knowledge on how Operating Systems work, as well as tailoring and configuring a Linux system is through building Linux From Scratch[1].

For my hardware, I chose the Compaq Presario X1000 laptop[2]. A laptop that has been very successful in the market, and has a lot of users whom wish to use Linux, but are either afraid of the move from another OS, or do not have the time to configure it by themselves. Also, at the time of entering into my 3rd year of study, and deciding what project to choose, this laptop was one of the high end laptops, and that gives me an advantage because I need good quality hardware, and a fast and powerful processor to compile a while Operating System from scratch.

I will explain the benefits of using a Linux Operating System, and move onwards to build a system from scratch to suit the laptop. I will then explain what problems a user may find, such as updating, securing, and maintaining the system, and give possible solutions to the problems.

I will also discuss the future of the project, how it could be further developed, and how the outcome project would benefit both the developer, and the user.

I believe there are many questions I should ask myself, and try to solve with the results and outcome of the project.

- How successful will this project be?
- How difficult is it to achieve my goals?
- How difficult is it for other developers to follow my path?
- Who are my target users?
- What future can I have with the outcome?

Hopefully I will manage to answer most of the questions by the end of the project.

1 Linux From Scratch - http://www.linuxfromscratch.org/
2 Presario X1000 laptop - http://h50025.www5.hp.com/hpcom/in_en/11_29_64_1659summary.html

Contents

Chapter 1 – Introduction **03**
Linux: What is an OS ? 03

Chapter 2 - The Problem **08**
What is in a Distribution? 09

Chapter 3 - Research & Common Scenarios **13**
Beginning Research 13
Scenarios 14
My Solutions
Alternative Solutions to the Scenarios

Chapter 4 - The procedure to building an OS **17**
The Preparation (8 stages) 17
The Preparation 18
Creating the partition & downloading the sources 18
Installing the 'basic development suite' 18
Entering the new system. Installing the base OS 18
Setting up the Bootscripts 19
BLFS – Installing the user applications. 19
Testing & Configuration 19
Building a home, Advertising my OS 19

Chapter 5 - The Building Process **20**
The Preparation 21
Creating the partition & downloading the sources 23
Installing the 'basic development suite' and the C Compiler. 25
Entering the new system, and installing the base of the OS 26
Setting up the Bootscripts 28
BLFS – Installing the user applications. 30
Testing & Configuration 46
Building a home, Advertising my OS 48

Chapter 6 - The future of this project **50**

Chapter 7 – Conclusion **52**

Appendix I – References **53**

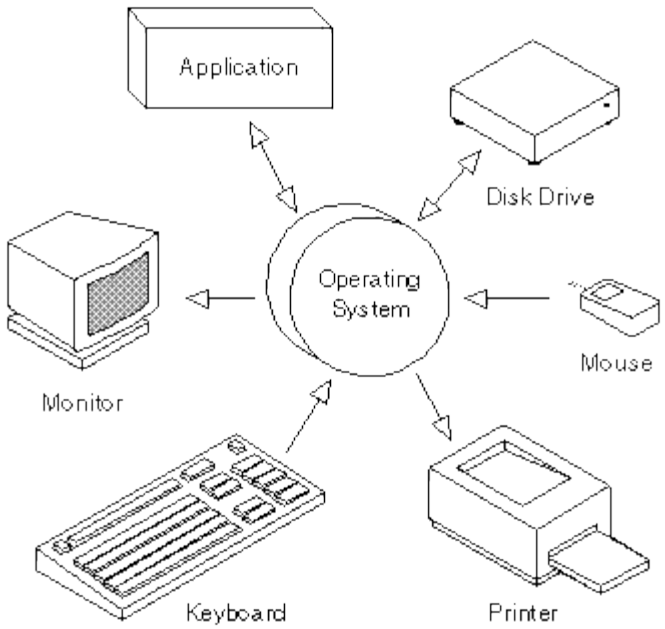
Appendix II - User's answers to the scenario questions on p.14 **55**

Appendix III - Images of Website for p.49 **59**

Chapter 1
Introduction

Linux: What Is an Operating System ?

[3]



An Operating System is a collection of programs and utilities, that manage the functions of a computer. It acts as a form of middle-ware between the software, and the hardware, linking them together. Operating Systems are divided into two sections.[4]

- The Kernel (or core)
  The Kernel is the heart of the OS. It will be defined shortly.

- The Applications
  Applications that make use of the computer, and allow the user to preform tasks and computations.

Linux is a free and open source Operating System, developed by Linus Trovalds. It is a clone of UNIX, developed from scratch to be compatible with the x86 CPU architecture, and later one evolved into

3 See References – Operating Systems
4 UHD - http://www.google.co.uk/url?sa=X&start=5&oi=define&q=http://www.uhd.edu/academic/support/lc/student_resources/glossary.htm

supporting systems running on Motorola, SPARC, RISC, and various other processors. Being free and open sourced, thousands of programmers, and engineers worldwide had the ability to download the source code, try it out, and further develop it, fixing bugs, and enhancing its security, power, and stability.

Linux, by itself, refers to the Kernel[5], which is the actual heart and fundamental part of the whole Operating System. It is a form of 'middle-ware' technology that provides the link between the software applications, and the underlying hardware. Its usage is limited until one combines other open sourced applications to do certain tasks a user needs.

When people of all ages, technical knowledge, and regions began to use Linux. They had the ability to switch their machine from a single user desktop, to a multi-user networked server which introduced Web Servers, Email Servers, Bulletin Board Systems and Forums, Firewalls, and many more services easily. However, with the use of such services came responsibility in assuring that the services, as well as the operating system itself remains secure enough to be trusted and used in the world wide web.

There are many alternatives to using Linux for both a general and a power user. However there are certain drawbacks I would like to comment on regarding each of them (Linux included)

Unix – BSD, Solaris[6]

Unix Operating Systems are very powerful. However, they can be very difficult to use. A user needs to learn difficult Unix fundamentals, and be prepared to configure every hardware by himself. Furthermore, modern hardware compatibility is an issue.

Windows 9x[7]

Windows 9x is based on very old technology. It is proved to be unstable and insecure. Furthermore, it is a single-user Operating System.

Windows NT (2000, XP)

Windows NT has come a long way. However it has more than once been brought down to a halt due to security holes and

5 The Kernel – http://kernel.org
6 See References - Unix
7 En1heure - http://www.en1heure.com/en/avantages_inconvenients_windows.php

insecurity. Being both a famous and buggy Operating System, it has attracted a large variety of viruses and spyware programs that further lead to instability, data corruption, and information theft.

However, one could argue that precautions, knowledge, updates, and proper use of firewalls and anti viruses could make NT stable and secure enough. There is still the very fact that Microsoft operating systems and applications are terribly expensive. As a user, not everyone can afford spending large sums of money to buy a licensed copy of Windows, and Office. As a developer, not everyone can afford spending thousands of pounds on developer suits such as Visual Studio[8].

So, the user has two possible outcomes in this situation. Either spend a lot of money for the products, and licensing, or revert to using pirated illegal versions.

Mac OS X

Although OS X is a powerful and user friendly OS, it has limited software compatibility, and it is used more as a user Operating System, rather than both a user and server OS.

Linux

One argument that people generally talk about is that Linux lacks standardization. This can be a good and bad thing. When researching about the topic, i managed to get several good remarks from LinuxQuestions.org[9].

My Question:
Its good that there aren't fixed standards. Choices are good.

Graphical Interfaces: KDE <> Gnome <> Xfce
Instant Messaging: Kopete <> Gaim.

8 Ebuyer - http://www.ebuyer.com/customer/products/index.html?rb=7193869993&action=c2hvd19wcm9kdWN0X292ZXJ2aWV3&product_uid=55133

9 Linux Questions - http://www.linuxquestions.org/questions/showthread.php?s=&threadid=318121

But could you agree that it sometimes causes problems? For example, a difference between each distribution is in the path they choose to install applications, which can sometimes lead to broken libraries, or “dependency hell”.

If you put it the other way around, is standardization an issue, or threat to Linux? both in the big picture, and the small one.

There must be at least **some** form of a standard, for example, FHS. The question really is, where does the standardization <> de-standardization point meet? And the fact that it is a gray area allows for circumstances to slightly push one way or the other sometimes.

If we do not have standards such as the File Hierarchy Standard (FHS), then imagine the mayhem Linux would be in. Now could we say the same thing with other standards?

For example, ff we had a standard for the graphical interface. Instead of having various types such as X11, Xorg, and Xfree competing against each other (note: competition can sometimes be good), would we achieve the same results, or better, if we had just **one** to work with?

The way I look at it, having it one way or the other might be bad. A mixture of both is good. The replies i got generally reflect on my point. And I quote,

gbonvehi

“ standards in Linux is almost impossible, it's simple, you have the oportunity to choose, and you choose acording to your tastes.
And of course, it depends where you look at this, it can be a good or bad thing.”

tangle

“*the FSH was a good idea that would have been implemented. It would of helped out everyone around. I do not mean that it is FSH way or not way, but I would like to see things install in the same place in every distro.*

Standards are good for the most part, imagine the internet if there where not standards.”

masonm

“As for standards, that almost flies in the face of what Linux is all about: choice. Everyone has the freedom to choose exactly what they want on their system, how it should work, and where to put it.

As long as there are people out there who think they have a better way of doing things that freedom is going to get exercised regardless of any arbitrary standards someone may wish to impose. This is the very reason Linux has progressed as far as it has.

While each distro set's it's own standards, it would be very difficult to establish any sort of community-wide standards for "these files should always go here" types of things as the type of people who are drawn to Linux, especially developers, are "do my own thing" kind of people anyway.”

Regardless of standardization issues, I believe that Linux is my preference of choice. It is free, open source, and has a very large community that can give a helping hand whenever a problem occurs. Linux is also very stable, secure, and perfect as an Operating System that has a balance between user based and server based systems.

Chapter 2
The Problem

Operating Systems have evolved quite a lot over time. However, they are not perfect, and still cause problems some of the times. The problems can vary from complicated tasks such as re-installing a corrupted system file, or identifying and installing a newly plugged in sound card. The more simpler tasks that a user may face vary from changing the screen resolution, to installing a mail program and sending an email over the Internet.

Although some of the problems a user faces shouldn't be there in the first place, while others may suddenly arise in certain situations. Either way, some basic knowledge of Operating Systems is needed to operate a computer, and the more knowledge a user has, the easier it will become to use, maintain, and fix a problem in a computer.

It is known for some time that Linux requires prior knowledge in computers and Operating Systems, and even the most important task of installing the Operating System (OS) onto the computer can be confusing, and result in multiple tries until the user is satisfied with the installation. Some Linux distributions tried to solve the problem[10], but drawbacks emerged where the ease of use sacrificed performance, as well as efficiency. If a hardware was not supported, it would be very difficult to bring up. If a hardware or software breaks, it would be very difficult to fix.

This problem is widely found in laptops. It can be a tedious job sometimes to have Linux running properly and stably on laptops straight out of a fresh installation. Users can sometimes be put off from the whole idea of using Linux, and revert to the OEM versions of Windows OS that came with the Laptops.

This is where XushiOS comes in handy. The OS will be compiled and built specifically to suit this laptop's needs. It will include all the necessary system applications and utilities, pre-compiled as well as configured to support all the laptop's hardware, and possibly other important hardware you might add, such as a USB mouse or Printer, thus resulting in more time used for the user's needs, rather than making the OS work.

10 XandrOS – http://www.xandros.com/
Linspire – http://www.linspire.com

What is in a Distribution?

The main question I would have to ask myself is,

what is so different between XushiOS, Slackware, and any other Linux distribution?

I have asked the question along with another question regarding standardization in a professional Linux forum. I have quoted some of the important key issues throughout the report.[11]

To summarize, Linux is Linux, as long as you are using the Linux Kernel. As for the rest of the applications, they are more or less the same in any distribution out there. I could see the exact same file hierarchy, and choice of packages in FreeBSD, Solaris, and Linux, but the **main** thing that differentiates them from each other is that Linux is using the Linux kernel, while FreeBSD is using the BSD kernel.

The only four possible differences I can see between Linux distributions include

- File system Hierarchy Standard (FHS), and how close or far they stick to it
- Package management, and how different they might be from one or another.
- Configuration, and how each distribution's developers add their own configurations by default.
- And finally, the destination where programs are installed.

But is that really it? Ignoring the legal terms of the actual name, can i call a distribution I am using now a "Slackware clone" just because it is FHS compliant, and uses TGZ format for packages? Can i create a Linux system from scratch, and call it a “Debian clone”, or a derivative of Debian if it complies with its file system hierarchy and supports the .deb package format? Can i call my distribution “XushiOS” just because it too follows the FHS standard, but uses my own formatting for creating packages? If I change the structure of my Operating System a bit so that it's structure matches Red hat, will it be called a “Red hat clone”?

11 Linux Questions - http://www.linuxquestions.org/questions/showthread.php?s=&threadid=318121

Or is it just the fact that Linux is Linux, being the kernel, and a distribution is just a *_reference to the developer's own work_* name given by one or a group of developers who save you the trouble, and configure it their own way? (For example, a computer is a computer. You can build it, but you have Dell, HP, that will do that for you.)
These are some of the replies i managed to receive, which address some of the issues mentioned above. The answers generally refer to Slackware distribution, because it is – similarly to XushiOS - the closest distribution to true Linux. And i quote,

Masonm

"File structure, "helper apps", auto-configs, and other tools added are what makes an actual distro. What makes Slack a little different is the simplicity of design, a noteable lack of "helper tools", very little auto-config, and a file structure which closely resembles the Unix system.

Also, Slack uses Unix type init scripts where many (most) distros don't. Slack's Keep It Simple approach makes it somewhat different that the more complex "automatic" distros that are becoming popular, especially with new Linux users.

Slack also doesn't alter the packages thus avoiding adding new bugs. The packages are pretty much the way the upstream developers designed them without Pat doing any additional tweaking to them. Most distros these days alter the packages, sometimes a great deal.

The packages also install into the directories the original developrs intended, not the way the distro developers decided they should. This is a major problem for some distros and can lead to dependency hell when some package manager is looking for a dependency where it should be, but because of distro changes, it's installed someplace else. This becomes an even bigger problem when trying to compile an app that looks for things where they are supposed to be while compiling.

Another major difference between distros in general is the way the packages are compiled. What compile options/flags are used, etc. This dramitically affects how the apps will run.

But yeah, all in all, Linux is Linux."

Tangle

"Now for the naming, Linux is the kernel. The rest, such as, FSH, apps and package manager is what defines a distro for the most part. If you where to make a distro from scratch, but base it (such as package managers) on a distro. I would say give that distro credit, but do not name it the same. That in my opinion is wrong. People put alot of work into making a distro, they deserver to have the name as their own. You also have to watch copy right laws if the name is copy righted. I find it nice that someone creates something and offers it for free. But they do have the right to keep somethings from themselves.

Some might differ and say all things should be free, but then again is anything really free? There is price to pay for anything that you do or aquire. Whether it be time, money and/or sweat. Nothing is free, but the best things are always earned."

Neo

"... a full install of the system doesn't come to a screaming halt like a lot of distro's. Almost nothing except the essentails are running by default. The rest is up to you to configure to your liking. There is almost no customization except where absolutely necessary. This makes it very different from just about every single distro I've ever used. In my opinion all these reasons make it more stable than anything else out there.

The following documentation describes how a developer can use LFS to build his very own custom Operating System, to achieve the goals previously discussed. For my example, I will show the development of XushiOS.

Chapter 3
Research & Common Scenarios

Beginning Research

In order to have a good feel of what I should be building, and what users expect out of an alternative questions, I decided to ask the users a few questions regarding the matter. The questions asked include,

1. What do you expect out of an Operating System

2. What disadvantages do you currently see in your OS? (security, viruses, updates, etc..)

3. If you had the opportunity to try Linux, what reasons would set you back or make you think twice before making the move?

4. What is your opinion on a Linux OS which is configured for your system, and would not require much configuration or setting up?

5. Do you like manually looking for updates on each and every application you have?

Some of the answers I received are noted at the end of the report. I did not include all the users I have asked, because it would greatly increase the number of pages of this dissertation.

Scenarios

I have created these scenarios, which are based on the questions answered by users. They reflect the users' opinions on their current Operating System, and what they expect a Linux OS should provide.

Scenario 1

A user bought a new laptop, but is unsatisfied with what the default Operating System gives him or her. Windows XP comes with little software, and if the user wishes to obtain Office applications, such as Microsoft Office, or developer tools such as Visual Studio, it will end up costing a lot of money.

.[12]

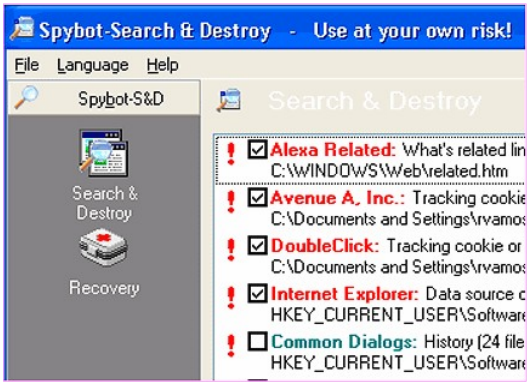
Solution

XushiOS is a good solution, because it is a free and open source OS. It comes with free built in office and developer tools, such as OpenOffice, and TCL/TK.

12 Image courtesy of Ebuyer – http://ebuyer.com

Scenario 2

A user is unsatisfied with how viruses and spy ware infest the OS, and lead to data corruption and data loss. The user also is afraid that the OS might be insecure, and sensitive data might be stolen.



.[13]

Solution

XushiOS is based on Linux, and research[14] shows that there are no viruses dangerous enough to damage a Linux OS. It has a strong admin and user based structure, where if any intentional or non intentional damage would occur, only the current user's home folder would be affected, and not the whole system itself.

Scenario 3

Users comment that it is tiring to manually look for updates for every application they have on their system. It requires the user to manually go to each application's website, check and download the updates. This can be very time consuming. Some of the users that currently use Linux find it difficult to install or upgrade software, let alone configure.

Solution

XushiOS uses a packaging system, which packages an application into

13 Image courtesy of ZDNet - http://reviews.zdnet.co.uk/software/utilities/0,39024208,10002916,00.htm

14 See references – Linux Viruses

a single file, and stores it in a centralized website. All the user has to do then is frequently visit this single website to check if there are updates. The advantage of having packages is that the user can install, upgrade, or remove packages with a single click, or command.

Currently, XushiOS uses two package methods. Autopackage[15], and Slackware's TGZ packages[16]. I am still undecided on which one to keep.

Alternative Solutions

There are of course alternative solutions to using XushiOS.

- Users may wish to install virus/spyware scanning and removal software to aid in prevention and recovery.
- Free, Shareware, or Open Sourced software may be used as an alternative to proprietary or expensive software.
- The user has a choice to use other operating systems, such as Apple's OSX, Sun's Solaris, or various Linux distributions if unsatisfied with the default Windows XP OS that comes with the machine.

However, I try to combine and give the users as many of the solutions listed above into one package or product, being XushiOS. And this is a very good advantage I have.

15 http://autopackage.org
16 http://www.slackware.com/config/packages.php

Chapter 4
The procedure to building an OS

The objective here is not only to create a Linux OS, but also to learn about the inner workings of Linux; The structure of the Operating System and how it works, the booting mechanism and method of loading critical applications into RAM, as well as the layout of the Operating System and what it does with its critical files.

To completely build my Operating System, and make it ready for beta testing or beginning use, I am told by the LFS book[17] that i would need three steps, or sections. However, i would like to further partition it down into **eight** major steps to be taken.

1) The preparation

To begin my project, I first need a host Operating System that will provide critical and necessary applications needed to start the process of creating XushiOS. These applications include a C/C++ compiler and linker to build the new system's compiler and base programs. I also need a working shell (command prompt) to do my work in.

The host Operating System of my choice is Slackware[18]. It is the first Linux distribution to come out, right after Linux 0.1. Slackware is the most Unix-like distribution there is, educating its user about the true nature of Linux and how it works. In my case, Slackware is very simple, stable, and straight forward.

Key advantages of Slackware over other distributions

- Patrick Volkerding, the lead developer, manages to use official and stable versions of all the tools needed in Slackware.[19]

- Patching programs or the kernel in Slackware is done only when it is absolutely necessary. This prevents bugs and increases the stability of the system as a whole.

---

17 LFS I.1 – Introduction - http://lfs.scarlet.be/lfs/view/stable/chapter01/chapter01.html#ch-scatter-how
18 Slackware – http://www.slackware.com
19 Answers.com - http://www.answers.com/topic/slackware

- Slackware gives the ability to have a minimal install of the Operating System, reducing bloat.

2) Creating the partition & downloading the sources

The next step is to create a partition for XushiOS to be compiled, and installed in. Afterwards, I will download the sources and patches needed to build the system.

3) Installing the 'basic development suite' and the C Compiler.

In this section, i will install the packages that are needed to form a basic development suite. This suite will be used in the next steps to build the actual system. Some of the key programs include the GNU C compiler, to compile the system, and Binutils, a package which contains a linker, an assembler, and other tools for handling object files.

4) Entering the new system, and installing the base of the OS

When completing step 3, I will have a temporary environment i can use to produce a clean build of my Operating System.

"The chroot (change root) program is used to enter a virtual environment and start a new shell whose root directory will be set to the LFS partition. This is very similar to rebooting and instructing the kernel to mount the LFS partition as the root partition. The system does not actually reboot, but instead chroots because creating a bootable system requires additional work which is not necessary just yet. The major advantage is that "chrooting" allows the builder to continue using the host while LFS is being built. While waiting for package compilation to complete, a user can switch to a different virtual console (VC) or X desktop and continue using the computer as normal."[20]

I will install the basic applications needed in order to have a working prototype of XushiOS.

20 Quote from LFS I.1 – Introduction - http://lfs.scarlet.be/lfs/view/stable/chapter01/chapter01.html#ch-scatter-how

5) Setting up the Bootscripts

When done, I can install the bootscripts that are needed to boot the Operating System. Afterwards, I will reboot into my new system.

6) BLFS – Installing the user applications.

Now that I have a working OS, i can start installing the applications a user would need, such as networking and sound support, a graphical user interface (Xfce), and several services (apache, MySQL).

7) Testing & Configuration

Last minute testing and configuration are to be done now. I will configure apache, MySQL, PHP, as well as set up the network, and the graphical interface.

8) Building a home, and Advertising my OS

Finally, all that is left is to create a home for my Operating System, and to advertise its existence. A website will be set up on my server under my domain[21].

21 Xushi – http://www.xushi.co.uk

Chapter 5
The Building Process

I will now enter the building stage. This section is where the actual work for building XushiOS begins. I will discuss each stage in brief. The actual commands and steps are shown in both the LFS and BLFS books.

LFS v6.0
http://lfs.scarlet.be/lfs/view/stable/

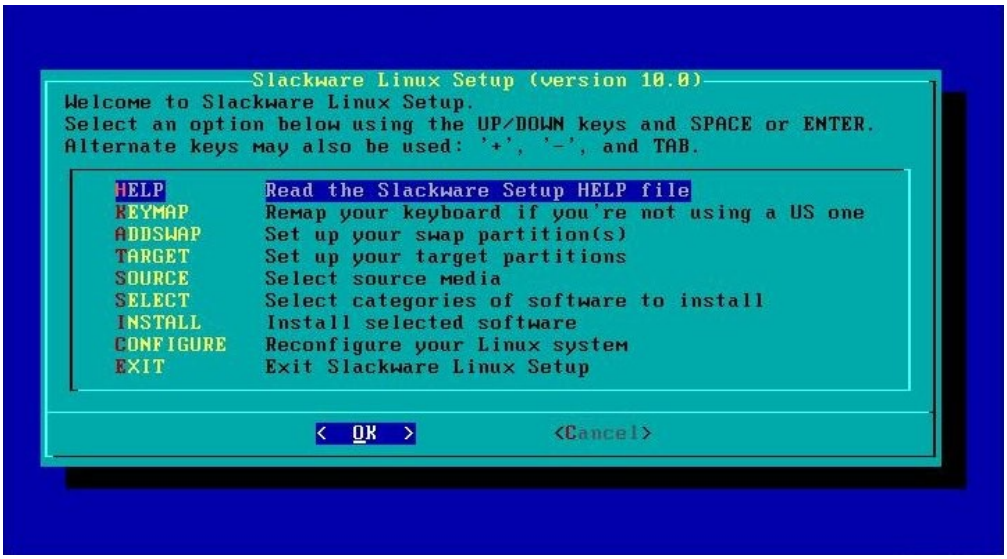
BLFS v6.0
http://lfs.scarlet.be/blfs/view/stable/

For this chapter, I must start with the LFS book. It is a very thorough step-by-step instruction book that guides the user on all the sources, procedures, and instructions needed in each stage. Since there are a few hundred programs in total that will need to be installed and configured, it would be better for me to show and example or two on each stage, summarizing what will be done throughout the whole process.

If you wish to see how specific programs are compiled, or what exactly i would do in any stage, the LFS and BLFS book's index page will have a link to each section and program. Similarly, for a full detailed view of the work to be done, the LFS and BLFS books could be referenced upon. Since I am legally allowed to, I will include a copy of the books along with my work on a CD.

Stage 1
The Preparation

Host OS: Slackware Working Shell: bash


Slackware Linux Setup (version 10.0)
Welcome to Slackware Linux Setup.
Select an option below using the UP/DOWN keys and SPACE or ENTER.
Alternate keys may also be used: '+', '-', and TAB.
HELP Read the Slackware Setup HELP file
KEYMAP Remap your keyboard if you're not using a US one
ADDSWAP Set up your swap partition(s)
TARGET Set up your target partitions
SOURCE Select source media
SELECT Select categories of software to install
INSTALL Install selected software
CONFIGURE Reconfigure your Linux system
EXIT Exit Slackware Linux Setup
< OK >
<Cancel>

Installing Slackware was an easy task which took less than an hour. Coincidently, it is my main OS. For more details, there are a few good tutorials on how to get the system up and running in a short hassle-free time[22].

Being a Linux user, I have a very good advantage. I can take some of the configurations of this system, and use them in XushiOS. This will save a lot of time later on, and will result in a cleaner and more stable OS, because I have already spent time configuring Slackware.

Some of the configurations i managed to use are

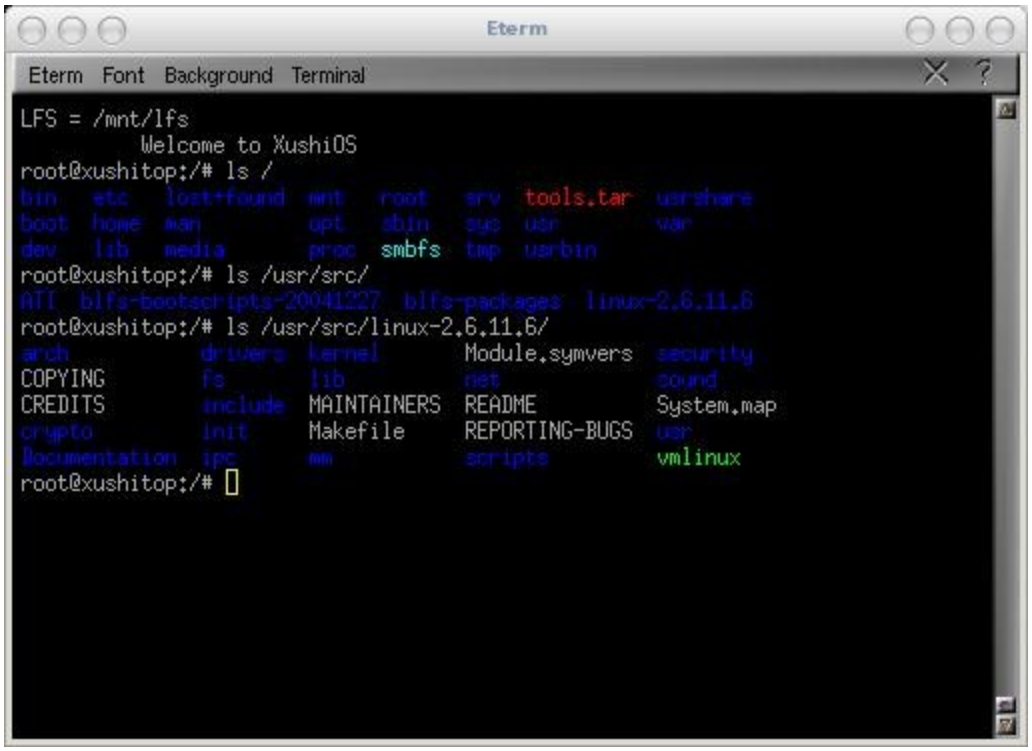
- Kernel 2.6 configs
  This allows me to use the very latest kernel on XushiOS. This is a very good advantage to other distributions since not all include kernel 2.6.

- Xorg config
  Xorg is the X window system, the core of the graphical interface for Linux. Since it is used in most distributions, i can safely reuse its configuration file for XushiOS.

22 See References – Slackware Installation.

Bash

Bash is the GNU version of the Unix shell, or command prompt. For my project, I use Eterm[23], a graphical front end to bash, so that i can open several instances of bash while on my graphical interface, to aid in simplifying the compilation process.



Additional programs used include ksnapshot[24], to take snapshots of my system for the project, and Gimp[25], the GNU Image Manipulating Program, to edit the screen shots i take.

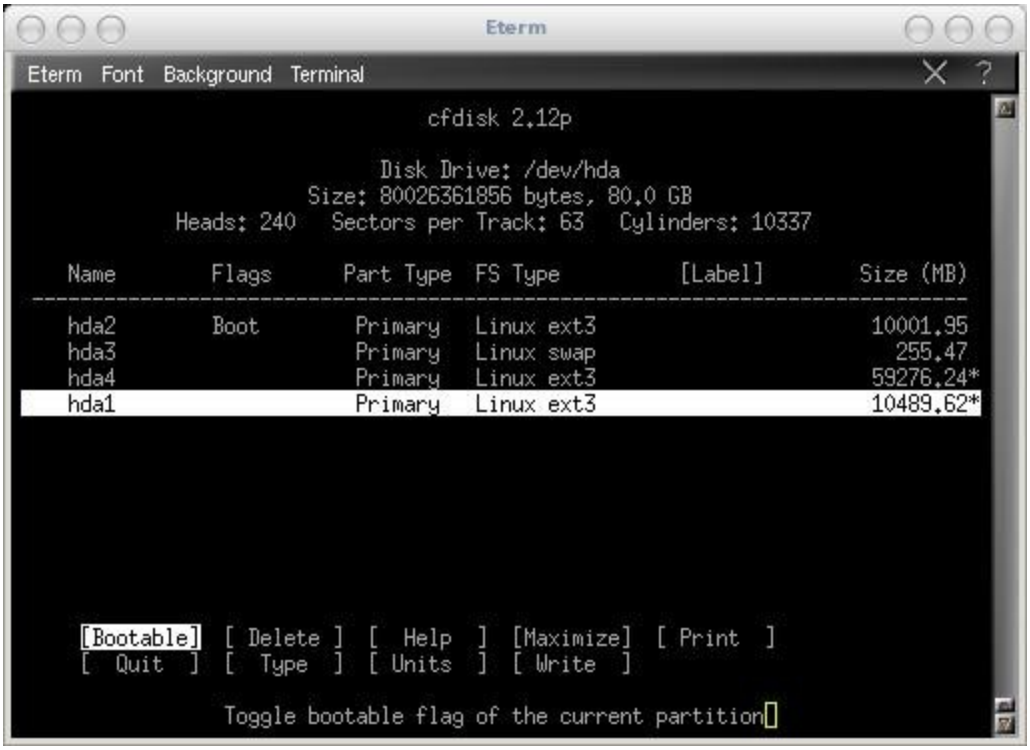
Additional preparations include creating an LFS user in the Host OS, and setting the environment and PATHs (shortcuts)

23 Eterm - http://www.eterm.org/
24 Ksnapshot - http://www.ipso-facto.demon.co.uk/ksnapshot/
25 http://www.gimp.org/

Stage 2
Creating the partition & downloading the source

Hard disk space is needed to build the system on. I have a 10 Gigabyte empty partition which I will use. Although building the base system will only require 3 Gigabytes, I left room for additional software to be installed, as well as temporary space for the source code and compiling.

Firstly, I need to create a new Linux partition. This is done through the **cfdisk** program. As shown below, I have created a 10.4 Gigabyte partition on /dev/hda1. Notice it says that the File System is currently ext3. This is not done yet, but it is shown below because my system is already set up.


Eterm
Eterm Font Background Terminal
cfdisk 2.12p
Disk Drive: /dev/hda
Size: 80026361856 bytes, 80.0 GB
Heads: 240 Sectors per Track: 63 Cylinders: 10337
Name Flags Part Type FS Type [Label] Size (MB)
hda2 Boot Primary Linux ext3 10001.95
hda3 Primary Linux swap 255.47
hda4 Primary Linux ext3 59276.24*
hda1 Primary Linux ext3 10489.62*
[Bootable] [Delete] [Help] [Maximize] [Print]
[Quit] [Type] [Units] [Write]
Toggle bootable flag of the current partition

To create the ext3 file system, i use the mkfs.ext3 command, which will create a journalized Linux file system for me.

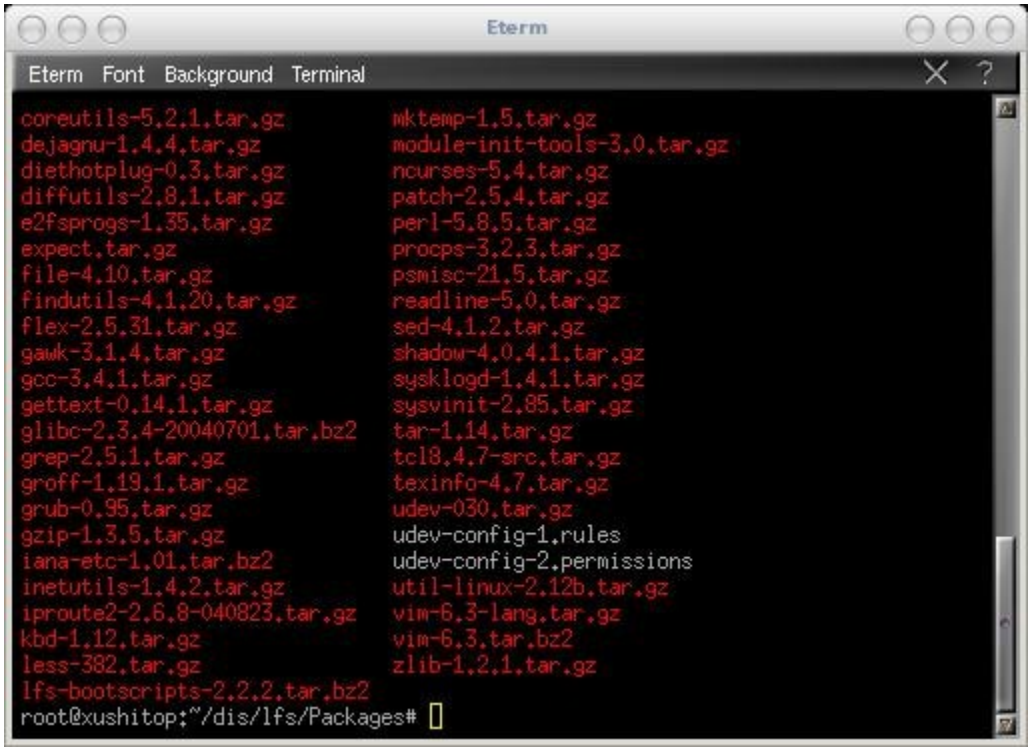
All i have left is to mount my new partition into a folder, /mnt/lfs, so that i can easily access it from within my host OS.

mount /dev/hda1 /mnt/lfs

What is left now, is to download all the needed packages and patches. I have a choice of either downloading them all at once[26], or manually download each source package when i need to work on it. In my case, since i have the hard disk space, i downloaded them all.

For a list of the important source packages needed, here is a link, http://lfs.scarlet.be/lfs/view/stable/chapter03/packages.html

Once all the packages are downloaded, i am ready to start the compilations.



26 LFS Sources - http://www.linuxfromscratch.org/lfs/packages.html

Stage 3
Installing the 'basic development suite' and the C compiler

The 'toolchain' is a set of programs of which include a compiler, assembler, linker, libraries, and a few useful utilities. The key note here is that both the Binutils and GCC packages are compiled twice, even three times. GCC always recompiles itself at least one more time

- A “first pass” GCC build is compiled using my host's original compiler and library files.

- A “second pass” GCC build is compiled with the previous 'first pass' compiler. This is the one that is installed. And GCC can be compiled for a third time to verify that the 'second pass' compiler can generate the same results over again.

- Afterwards, I can use that newly built toolchain to build the essential utilities needed for my system

When all of the base packages are installed, i can use the shell to enter into my new OS and its partition, and begin to use its toolchain to build the sysetm.

Stage 4
Entering the new system, and installing the base OS

The list of programs I had to install is very large. The guide on installing each individual application is documented in the Linux From Scratch[27] and Beyond Linux From Scratch[28] online books. However, i will show an example of how i compile and install most of the programs listed.

Compiling From Source

Here is a guide on how to compile from source. Basically, compiling a program is a three step procedure (plus a fourth extra step if needed).[29]

- ./configure
  This is a script which checks your system to make sure that it is equipped with all the needed compilers, libraries, and dependencies. Parameters could be passed to give the script additional information, such as a library path, destination path, or features in the program to enable/disable.

- make
  make is a utility which actually compiles the source, based on instructions it needs in the 'Makefile' – located in the source directory.

- make install
  when you run 'make' with the parameter 'install', it searches the Makefile for that particular label, and only executes that particular section. 'make install' installs the program by copying them to their destination directories.

- Additional commands if necessary
  Occasionally one needs to enter a few specific commands to complete the installation. Example, when installing libraries, one needs to type

27 LFS - http://lfs.scarlet.be/lfs/view/stable/
28 BLFS - http://lfs.scarlet.be/blfs/stable.html
29 See References – Compiling Applications From Source.

ldconfig

to update the library tree. Or when installing lilo (the linux loader), one would need to type

lilo

before rebooting.

Example 1

To install a simple utility such as 'Gawk', I will have to unpackage the source package, and compile it.

code:

```
tar xvzf gawk-3.1.4.tar.gz && cd gawk-3.1.4/

./configure --prefix=/tools

make

make install
```

Example 2

To install a more complex program such as GCC, it is the same procedure, but with a few extra parameters.

code:

```
tar xvzf gcc-3.4.1.tar.gz && cd gcc-3.4.1/

./configure –prefix=/tools --enable-local=gnu –enable-shared
--enable-threads=posix –enable-shared –enable-__cxa_atexit –
enable-languages=c,c++ --disable-libstdcxx-pch

make

make install
```

Stage 5
Setting up the Bootscripts

At this stage, the compilation is complete. I now have a base Operating System that i can use. However i need to setup and install the Bootscripts so that the new OS will have the ability to boot, and load up by its self.

Linux uses a standard based booting facility called **SysVinit**. It is based on the concept of using seven **run-levels**, each with its own special purpose. The standard states that the following levels are defined as,

Run Level	***Description***
0	Halt the computer (ready to be rebooted or turned off)
1	Single-user mode
2	Multi-user mode without networking capabilities
3	Multi-user mode with networking capabilities
4	Multi-user mode with a graphical interface
5	Reserved
6	Reboot the computer

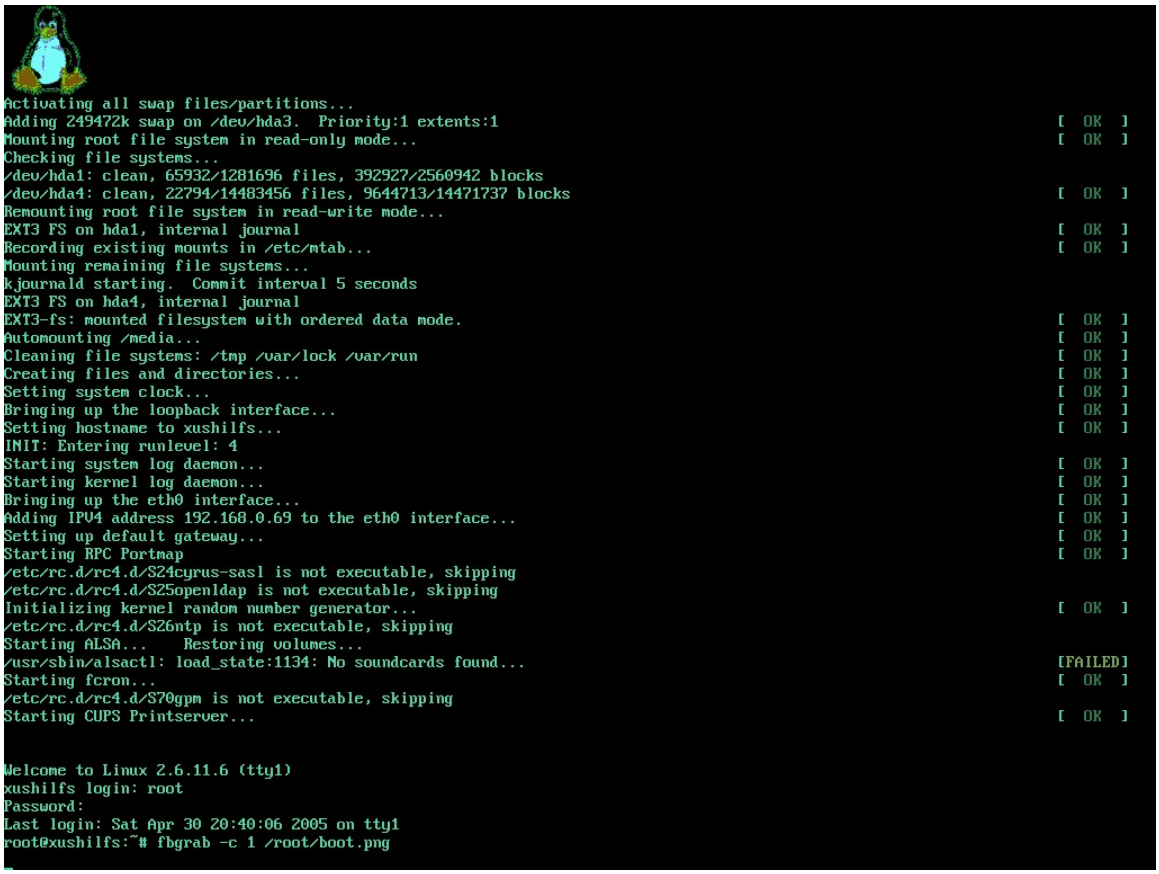
LFS prefers to use Unix System-V style init scripts, because they are widely used and available. To install the Bootscripts package, I just need to pass the

make install

command.

Other programs are also installed at this stage, including device handling programs (udev), and certain configurations such as the bash shell and networking scripts are installed.

Shortly afterwards, I managed to reboot my laptop and enter the newly built Operating system. A snapshot of the booting process is in the following page.


Activating all swap files/partitions...
Adding 249472k swap on /dev/hda3. Priority:1 extents:1 [OK]
Mounting root file system in read-only mode... [OK]
Checking file systems...
/dev/hda1: clean, 65932/1281696 files, 392927/2560942 blocks
/dev/hda4: clean, 22794/14483456 files, 9644713/14471737 blocks [OK]
Remounting root file system in read-write mode...
EXT3 FS on hda1, internal journal [OK]
Recording existing mounts in /etc/mtab... [OK]
Mounting remaining file systems...
kjournald starting. Commit interval 5 seconds
EXT3 FS on hda4, internal journal
EXT3-fs: mounted filesystem with ordered data mode. [OK]
Automounting /media... [OK]
Cleaning file systems: /tmp /var/lock /var/run [OK]
Creating files and directories... [OK]
Setting system clock... [OK]
Bringing up the loopback interface... [OK]
Setting hostname to xushilfs... [OK]
INIT: Entering runlevel: 4
Starting system log daemon... [OK]
Starting kernel log daemon... [OK]
Bringing up the eth0 interface... [OK]
Adding IPV4 address 192.168.0.69 to the eth0 interface... [OK]
Setting up default gateway... [OK]
Starting RPC Portmap [OK]
/etc/rc.d/rc4.d/S24cyrus-sasl is not executable, skipping
/etc/rc.d/rc4.d/S25openldap is not executable, skipping
Initializing kernel random number generator... [OK]
/etc/rc.d/rc4.d/S26ntp is not executable, skipping
Starting ALSA...   Restoring volumes...
/usr/sbin/alsactl: load_state:1134: No soundcards found... [FAILED]
Starting fcron... [OK]
/etc/rc.d/rc4.d/S70gpm is not executable, skipping
Starting CUPS Printserver... [OK]
Welcome to Linux 2.6.11.6 (tty1)
xushilfs login: root
Password:
Last login: Sat Apr 30 20:40:06 2005 on tty1
root@xushilfs:~# fbgrab -c 1 /root/boot.png

Note: The failure shown above refers to the sound mixer, which I can not find the sound card. I will enable it the next time I recompile the kernel.

Stage 6
BLFS – Installing the user applications.

By now, After three weeks of compiling well over 700,000 files, i have a working bootable Linux Operating System (hooray!). But a user cannot yet make much use of it, as it is just a base OS, with nothing more interesting than a command line, and ping. This is the time to move into the “Beyond LFS” book, and begin to install necessary applications such as a user interface, an Internet browser, and so on.

Again, the list of packages to be installed is extremely large, larger than the list found in LFS. I will show an example of both ways I use to compile and install most of the programs listed.

- Compiling from source
- Installing and configuring through pre-compiled packages.

Compiling From Source

For my example, I will compile the Apache2 web server[30]. Two parameters are passed to the configure script. The first one sets the destination folders based on the File Hierarchy Standard[31], while the second one enables shared mods.

The procedure is as follows

code:

```
./configure –enable-layout=FHS –enable-mods-shared=all

make

make install

chown -R apache:apache /usr/local/apache2
```

30 Apache 2 - http://httpd.apache.org/
31 FHS - http://www.pathname.com/fhs/

Pre-compiled Packages

Having pre-compiled packages is a good step-forwards for Linux. Up till a few years back, installing programs was easy. However, problems emerged when one wanted to uninstall or upgrade them. Not every program has an uninstall script, and not every uninstall script worked perfectly.

Having packages became a good idea. When a developer or user compiles a program, he would then create a package that contains all the compiled binaries and files, archived in their relative destination paths. When a user wishes to upgrade or uninstall the program, he then simply executes a script which would easily find all the necessary files, and removes or upgrades them. Examples of package formats include Slackware's TGZ, Debian's DEB, and Redhat's RPM packages.

Further development lead to packages managers. Now a user can browse through a list of installed packages, to select one or a group of packages to be installed or uninstalled. This greatly eases the installation and upgrading of software in Linux, and increases stability of the system, where as you wouldn't need to worry about dependencies anymore. Furthermore, repositories can be created, where a group of developers can share their packages to the rest of the community, thus helping out any other users who use the Operating System

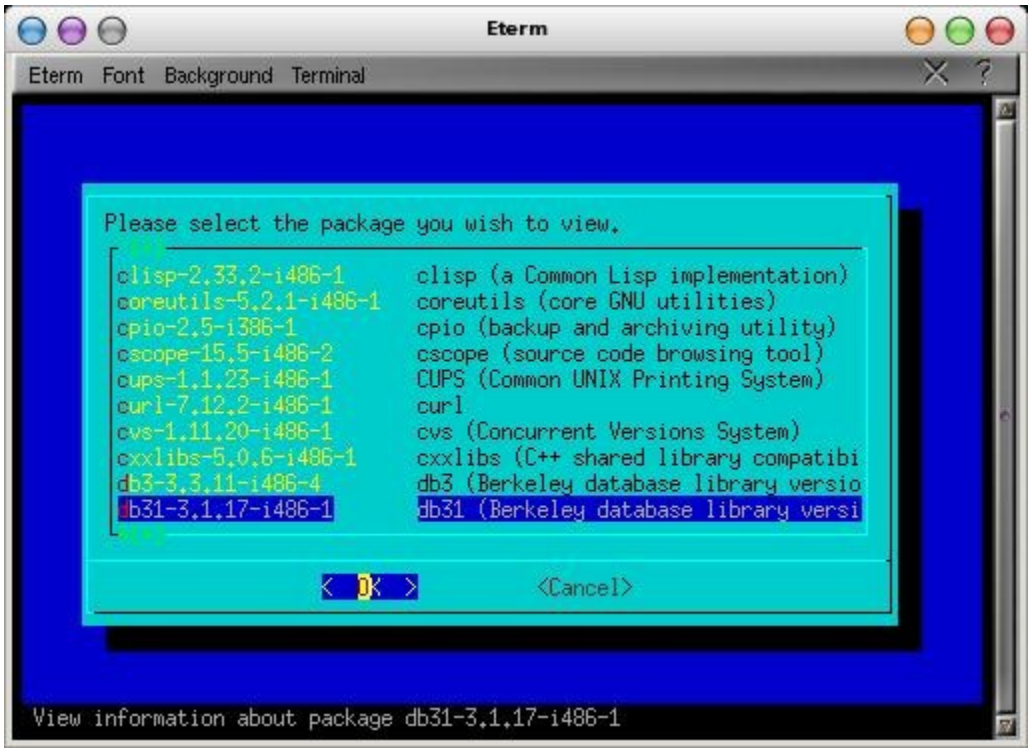
There are several different package managers to use. I have tested three of them.

- Slackware's Package Manager[32]
  Lightweight and down-to-earth. Its packages are simply made of a compressed archive.
- Paco
  Small and easy to use, but has poor package listing and package removal.
- Autopackage
  Very new. Gives the ability to install both from the command line, and the GUI.

I would have chosen to stick with Autopackage, but it only came out

32 http://www.slackware.com/config/packages.php

last month. I did not have enough time to test it, and see how truly stable and efficient it is. However, I did choose to use Slackware's package manager, **pkgtool**. The advantage it has over the other ones is that it uses a user friendly interface to list the packages, and give you a choice on what to do next.


Eterm
Eterm Font Background Terminal
Please select the package you wish to view.
clisp-2.33.2-i486-1 clisp (a Common Lisp implementation)
coreutils-5.2.1-i486-1 coreutils (core GNU utilities)
cpio-2.5-i386-1 cpio (backup and archiving utility)
cscope-15.5-i486-2 cscope (source code browsing tool)
cups-1.1.23-i486-1 CUPS (Common UNIX Printing System)
curl-7.12.2-i486-1 curl
cvs-1.11.20-i486-1 cvs (Concurrent Versions System)
cxxlibs-5.0.6-i486-1 cxxlibs (C++ shared library compatibi
db3-3.3.11-i486-4 db3 (Berkeley database library versio
db31-3.1.17-i486-1 db31 (Berkeley database library versi
< OK >
<Cancel>
View information about package db31-3.1.17-i486-1

To create a package, simply replace the command 'make install' with 'checkinstall'. Checkinstall will automate the creation of packages.

When done, you can share the package, or keep it for backup. Any user who downloads it can use any of these commands

- installpkg
- updatepkg
- removepkg

To install the program, update the currently installed program with the new package, or uninstall the program respectively.

Configuring, Part 2 – AMP & The Kernel.

There are two very important programs I need to talk about for my project. AMP (Apache, MySQL, PHP) and the Kernel 2.6. One advantage of having a Linux based OS is that you can turn any computer into a server in a matter of moments. If a Pentium 133Mhz machine can easily serve a website, then so can my laptop running XushiOS. That way, a user can easily post notes, pictures, and forums to the world from his laptop. But in order to have a secure website, I first need to have a stable up-to-date kernel.

I have written two full guides on how to install and configure both the latest 2.6 Kernel, and the AMP package in under 10 minuets. Notice how they use Slackware's TGZ package format. These tutorials show two key points.

1. The power of using packages to reduce the time needed to install software (used in AMP installation)
2. The simplicity of Linux. Contrary to what people think, Linux is easy to learn.

The Linux 2.6.11 Kernel

There are 4 stages to getting your 2.6 working
1- get your system ready for 2.6
2- download and unpack kernel
3- configure kernel
4- compile kernel

==========
STAGE 1: Getting Ready
==========

Before starting, we need to edit the system to make it compatible and ready for the 2.6 kernel. (2 steps)

Step 1

download these necessary packages and install them

udev (currently version 054)
http://slackware.com/pb/searchpkg.p...amp;string=udev

diffutils (currently version 2.8.1)

http://slackware.com/pb/searchpkg.p...amp;string=diff

module-init-tools (currently version 3.1)
http://slackware.com/pb/searchpkg.p...ing=module-init

Step 2

we need to make two backups of the kernel. The first one is this original kernel you have right now (the 2.4.29, soon to be obsolete), and the second backup is so that every time you compile a new kernel, you have the ability to go back to the last good compiled one in case anything went wrong.

if you ls /boot , you will see a vmlinuz-ide-2.4.29 . Keep this file name in handy..

now, edit lilo.conf

code:

```
pico /etc/lilo.conf
```

add an entry at the bottom, which would look like this

code:

```
image = /boot/vmlinuz-ide-2.4.29
    root = /dev/hda1
    label = SlackOld
    read-only
```

Note: replace /dev/hda1 with your root directory if needed.

And add the second entry like so,

code:

```
image = /boot/vmlinuz.old
    root = /dev/hda1
    label = Slack10.old
    read-only
```

That one is the last successful compiled kernel. Save, and exit.
Now we're ready to start the upgrade.

==========
STAGE 2: Downloading and unpacking
==========

1- Download the kernel.

Currently, the latest kernel is 2.6.11.7. Download it from kernel.org. The direct link is

code:

```
wget http://kernel.org/pub/linux/kernel/....6.11.7.tar.bz2
```

copy or download that file into your /usr/src/ folder
if you have a link in /usr/src called linux, delete it.

code:

```
cd /usr/src/
rm linux
```

unpack the kernel

code:

```
tar xvjf linux-2.6.11.7.tar.bz2
```

make a link to the new kernel and change directory into it

code:

```
ln -s linux-2.6.11.7 linux
cd linux
```

To save time after compiling, i like Linux to automatically move the new kernel into boot, and to update lilo for me. This is done like so,

Edit the Makefile, search for the line '#export INSTALL_PATH=/boot' and uncomment it.

code:

```
pico Makefile
ctrl + w  (type boot)
remove the leading #
save and exit.
```

Now we're ready to configure the kernel.

==========

STAGE 3: Configuring

==========

You need a .config file. You could either
- start from scratch and edit the new kernel, which is very long and prone to errors.. or
- use your existing 2.4.29's .config file (not really recommended), or
- use someone else's .config file as a basis, modifying and changing/enabling what you need.

To start from scratch, type

code:

```
make menuconfig
```

(if you're in a command line, or)

code:

```
make xconfig
```

(if you are running from a graphical interface, like KDE.. its easier on the eyes.)

If you want to use your 2.4.29 config, just copy it over

code:

```
cp ../linux-2.4.29/.config ../2.6.11.7/
```

alternatively, if you already have a 2.6.11.6 .config, you can use that (recommended, especially when you're upgrading kernels later on)

code:

```
cp ../linux-2.6.11.6/.config ../2.6.11.7/
```

The main thing to note is that you need a .config

now we're ready to compile our kernel.

==========

STAGE 4: Compiling

==========

all you have to do is type

code:

```
make
make modules_install
make install
```

have some green tea, and wait till its done.. That's it
When you're all done, just reboot. Congratulations :)

- -

Note: Keep your .config file handy.. back it up. Whenever a new kernel comes out, all you have to do is download the source, unpack, copy the .config over, and

code:

```
make
make modules_install
make install
```

and in 15 min, you have the latest kernel again.

AMP – Apache, MySQL, PHP

AMP is divided into three sections.

- Apache – An HTTP server
- MySQL – A database management system, in our case, used to store data a user will input into a forum on the website.
- PHP – A server-side HTML embedded scripting language used to create dynamic web content.

Notice: Instead of configuring and compiling from source, 'installpkg' saves us a lot of time and effort, reducing our troubles down to one single line of command.

Note: for a text editor, here i use pico. Here's a quick tutorial on how to use it, to edit a file

code:

```
pico myfile
```

code:

```
pico /etc/apache2/myfile
```

to save

code:

```
ctrl + o
enter
```

to exit

code:

```
ctrl + x
enter
```

to save and exit (together)

code:

```
ctrl + o + x
```

```
enter
```

==========
MySQL 4.1.11
==========

download from linuxpackages.net
==========

code:

```
wget http://ftp.scarlet.be/pub/linuxpack...1-i486-2mjg.tgz
```

install
==========

code:

```
installpkg mysql-4.1.11-i486-2mjg.tgz
```

create a group called mysql
==========

code:

```
groupadd mysql
```

create user mysql (if you don't have him.. to see if you do, type su - mysql. If it says "Unknown id: mysql" then you don't have that user.
==========

code:

```
useradd -g mysql mysql
```

give user mysql access
==========

code:

```
chown -R mysql.mysql /var/lib/mysql
```

install a database from user mysql
==========

code:

```
su - mysql
mysql_installdb
exit
```

create a my.cnf file
==========

code:

```
cp /etc/my-medium.cnf /etc/my.cnf
```

keep the socket in /var/run/mysql/, its more secure than /tmp.
don't modify anything else

start mysql
==========

code:

```
chmod +x /etc/rc.d/rc.mysqld
su - mysql
/etc/rc.d/rc.mysqld start
exit
```

set a password for mysql root user
==========

code:

```
/usr/bin/mysqladmin -u root password 'new-password'
```

From this point onwards, i prefer to use mysqlcc to set mysql up..
download it from

http://dev.mysql.com/downloads/other/mysqlcc.html

unpack it
==========

code:

```
tar xvfz mysqlcc-0.9.4-linux-glibc23tar.gz
```

run it
==========

code:

```
cd mysqlcc-0.9.4-linux-glib23/
./mysqlcc
```

file > new
name: mylocalmysql
hostname: localhost
user: root
password: **********
socket file: /var/run/mysql/mysql.sock

click on test, it should be successful

==========

APACHE 2.0.53

==========

download apache
==========

code:

```
wget ftp://ftp.scarlet.be/pub/linuxpacka...3-i486-2spi.tgz
```

install apache
==========

code:

```
installpkg apache2-2.0.53-i486-2spi.tgz
```

edit apache config file
==========

code:

```
pico /etc/apache2/httpd.conf
```

You don't really have to edit anything right now. Apache should work with the current configs.

==========
PHP 5.0.4
==========

download php- apache2
==========

code:

```
wget ftp://ftp.scarlet.be/pub/linuxpacka...4-i486-1spi.tgz
```

install php- apache2
==========

code:

```
installpkg php-apache2-5.0.4-i486-1spi.tgz
```

download & install t1lib 5.0.2 (needed)
==========

code:

```
wget ftp://ftp.scarlet.be/pub/linuxpacka...2-i486-2spi.tgz

installpkg t1lib-5.0.2-i486-2spi.tgz
```

configure php

edit httpd.conf and make sure this line is at the bottom of the file

code:

```
pico /etc/apache2/httpd.conf

Include /etc/apache2/mod_php.conf
```

edit mod_php.conf

make sure the line looks similar to this

code:

```
LoadModule php5_module lib/apache2/libphp5.so
```

start apache & php

code:

```
apachectl start
```

or

code:

```
/etc/rc.d/rc.httpd start
```

When started, go to your web browser, and go to

code:

```
http://localhost
```

you should see a website. Good.

let's test php

code:

```
cd /var/www/htdocs
```

create a file called test.php and add the following to it

[CODE/**<? phpinfo(); ?>**[/CODE]

You should now be able to see all the php configurations.
If you search for mysql, you won't find it.
Let us link mysql to apache and php now.

stop apache

code:

```
apachectl stop
```

or

code:

```
/etc/rc.d/rc.httpd stop
```

go back to /etc/apache2, and edit php.ini

code:

```
cd /etc/apache2/
pico php.ini
```

search for mysql

code:

```
ctrl + w  mysql
```

uncomment the line ;extension=mysql.so by removing the ';'
==========

code:

```
extension=mysql.so
```

restart apache
==========

code:

```
apachectl start
```

refresh the http://localhost/test.php
and search for mysql
now you will find it halfway through the web page.

Congratulations..

Stage 7
Testing & Configuration

This is a “last minute” stage, where I double check all my settings, make sure the scripts are working, and fix any problems I find. As the saying goes,

“Better safe than sorry”

In this stage,

- I modified the Xorg graphical interface's configuration file to suit my needs.
- I installed ATI's 3D acceleration drivers
- I edited Apache's configuration file to tighten up security.

Security System used in apache:

.htaccess (mod_access). The .htaccess files provide a way to make configuration changes on a per-directory basis.

Security for apache:

httpd.conf. Each directory to which Apache has access can be configured to decide on which services and features are allowed and which are disabled in that directory (and its subdirectories).

Example:

```
<Directory /foo>
Deny All
AllowOverride None
Allow from all
</Directory>
```

Alias: If you want more than one user to add their web pages to your server, you can assign a folder in their home directory where they can put their web documents in. That increases security and privacy to each user and the server location too. To do so with a user called john, you need to create a folder in his home directory called .. say, public_html, and add this line to your httpd.conf file.

```
Alias /john/ "/home/john/public_html/"
```

This security measure can also be used for scripts.

Security model of filesystem:

The security model of the UNIX file system is affected through 'chmod' chmod is a utility that changes the permission mode of a file. This can be handy if you create Aliases for users, and expect them to program in their own home folders. In the example used above, I should use

chmod 755 public_html

to allow the proper permissions needed for users to view the files in that directory. Also, when using CGI or PHP scripts, the users might need to read/write in current /cgi directory, or some other cgi directories in their own home dir. So I have to set up chmod permissions for apache users and also the read/write permissions.

How file system affects apache security:

chmod allows me to make sure that none of the scripts can gain the rights they are not suppose to have, or read/write in folders with sensitive and private information. This is especially important so that http://xushi.co.uk/~username is properly controlled by the user himself only. A malicious user will not be able to put dangerous scripts that could damage the machine or lose data.

Further references on how to secure and administer Apache include:

Authentication, Authorization and Access Control:
http://httpd.apache.org/docs-2.0/howto/auth.html

Per-user web directories
http://httpd.apache.org/docs-2.0/howto/public_html.html

suEXEC Support
http://httpd.apache.org/docs-2.0/suexec.html
The suEXEC feature provides Apache users the ability to run CGI and SSI programs under user IDs different from the user ID of the calling web-server. Normally, when a CGI or SSI program executes, it runs as the same user who is running the web server.

Stage 8
Building a home. Advertising my OS

At this stage, my work is done! I have a ready built Operating System, with a user interface, and a few applications, services and daemons.[33] All that is left to do now is to create a home for it, i.e. a web page, and to advertise its existence so that other users and developers can try and help maintain the system.

The Website

Since I already own a server and a domain, I decided to use them to host the web page. Creating it was very easy because i already have a background knowledge in PHP programming, and using PHP scripts eases the transaction of creating more than one page. The end result can be both seen in the blog[34], which will be updated with current news regarding XushiOS, and a section of the website dedicated to providing information regarding the Changelog, FAQ, and download area[35].

The current web page is located here,
http://xushi.co.uk/xushios/

Advertising

Advertising is an important step to take when developing an Operating System. There are already a wide variety of distributions to choose from, but my OS has a certain distinctiveness, or feature, which others may not have. It is specifically designed for this laptop, where as a user will install XushiOS in a matter of minutes, and little configuration will be needed afterwards since almost everything will already be configured by default.

The OS also has an advantage, where little modification to the original toolchain is needed in order to recompile a new Operating System for another model or brand of laptop.

---

33 See Appendix III for images.
34 XushiOS Blogk - http://xushi.co.uk/
35 See Appendix III for images.

However, my work is still not 'complete' enough for me to be satisfied enough to distribute it yet. I sill would like the opportunity to start over again, reducing it's size, and gaining more knowledge and perfection in operating and moving throughout the eight stages mentioned above.

When I do advertise the Operating System, I need to pick certain areas that

- Have a high probability that the user browsing owns the same laptop model XushiOS is based on,
- Have low or no cost for me to advertise.

The best places I found include,

- The Unofficial Compaq forum[36]
  This forum is dedicated for the X1000, NX7000, and ZT3000 modeled laptops. It is the perfect place for me to advertise in.

- Linux Questions Forum[37]
  The Linux Questions forum is the most used forum for questions and tutorials regarding the Linux and Unix OS.

- Review Magazines.[38]
  Each month, certain Linux magazines issue reviews of new Operating Systems that have come out. I believe they would give a good review, especially to the fact that my distribution concentrates more on Laptops.

36 X1000 Forum - http://www.x1000forums.com/
37 Linux Questions – http://linuxquestions.org/
38 Linux Format - http://www.linuxformat.co.uk/
Linux User - http://www.linuxuser.co.uk/

Chapter 6
The Future

I see three outcomes of this project

- Knowledge
- Custom LiveCD and Installer
- Custom Linux Operating Systems for laptops.

Knowledge

The LFS project provided me with a lot of information that taught me the inner workings of Linux. I have learned more about how Linux operates, boots, what it is comprised of, while developing LFS in a month's time, than using Linux and reading tutorials and books in three years.

I have managed to create a web blog in my site, and combined my own website with XushiOS's page. Future developments include,

- Creating a package repository, so that my users can just install software with a single click, instead of the hours spent compiling and configuring. (Think of the repositories found for XandrOS[39], Linspire[40], or Slackware[41])

LiveCD

A LiveCD is a very good idea for any user who wishes to try out the OS, but not overwrite anything on their current system. This way, the user can try Linux, then when done, eject the CD and reboot, and the user is back to his original system. Hopefully, if this project becomes successful enough, I could release XushiOS as a LiveCD, or even an installable Operating System, where anyone who has a Compaq HP Presario X1000 could just insert the CD, install the OS with a few clicks, and have everything working in a matter of minutes.

---

39 Xandros repo. - http://www.xandros.com/products/desktop_matrix_apps.html
40 Linspire repo. - http://www.linspire.com/lindows_products_categories.php
41 Official: http://slackware.com/pb
Unofficial: http://linuxpackages.net

Custom Linux OS

If the project is successful enough, I could easily convince companies such as HP, Dell, and other manufacturers or retailers to include LiveCDs or OEM versions of my Operating Systems on their laptops. It would a cheap, and very easy method of introducing the mass to Linux. The advantage here is that i will be doing all the work in ensuring compatibility, and that the system would work 100% off the CD with whatever laptops they inquire about. They will end up with a fully configured Operating System on a CD, with no hassle in building, maintaining, or technical support.

Chapter 7
Conclusion

This has been a very interesting and challenging project. I have gained much knowledge on the technicality of Operating Systems, Linux, and on how to write up a nice and long report. I began with a situation where I needed to implement and design a solution to my problem, and I resulted with a custom built Operating System, and an introductory website for it.

Hopefully when the time comes, I will have the ability to expand on the project, and release it for public use.

The key points to show here is that I now provide the ability to ease the moving into Linux, through a custom built pre-configured Distribution. My problems were based on how both to allow a user to easy use a Linux Operating System on a laptop, and go through the process of building it.

As said before, Linux is Linux. It may seem difficult to use, and the constant switching to the command prompt may frighten users at first, but you can not expect to use an OS without knowing the basics of it. It is similar to using windows without knowing how to install a program, or driver, or how to change resolutions.

It is not difficult to use a Linux system, regardless of which distribution. And as I have shown in this dissertation, it is not difficult to build your own distribution from scratch.

Appendix I
References

Operating Systems

Image taken from
http://www.webopedia.com/TERM/o/operating_system.html

The Linux OS

Thinkquest -
http://library.thinkquest.org/C003740/history.html
https://netfiles.uiuc.edu/rhasan/linux/

Linux International -
http://www.li.org/linuxhistory.php

Slacklife -
http://www.slacklife.com.br/article.php?sid=1063

Unix

Mac OSX Hints -
http://www.macosxhints.com/comment.php?mode=display&sid=20040429091024304&title=Disadvantage+of+UNIX&pid=0

UWSG -
http://www.uwsg.iu.edu/usail/external/recommended/unixhx.html

Rate My Web Host -
http://www.ratemywebhost.com/faq/basics/unix-web-hosting.html

Linux Viruses

Librenix -
http://librenix.com/?inode=21

The Register -
http://www.theregister.co.uk/2003/10/06/linux_vs_windows_viruses/

Slackware Installation

Slackware Linux Essentials Book -
http://www.slackware.com/book/

Shilo's Slackware guide
http://shilo.is-a-geek.com/slack/

Compiling applications from source

Start Linux -
http://www.start-linux.com/articles/article_171.php

Geocities, Tips for Linux -
http://www.geocities.com/tipsforlinux/articles/27.html

Books

Walnut Creek (1999) Linux The Complete Reference, 7th ed. John Purcell. ISBN: 1571761993

Walnut Creek (1999) Linux The Advanced Reference, 7th ed. John Purcell. ISBN: 1571762493

Wiley Publishing (2005) Beginning PHP5, Apache, MySQL Web Development. Naramore, Gerner, Scouarnec, Stolz, Glass. ISBN: 0764579665

Wiley Publishing (2005) Professional PHP5. Thompson, Goodman, Nowicki, Cove. ISBN: 0764572822

BSDi (2000) The Official Guide to Slackware Linux: Slackware Linux Essentials. 1st ed. Cantrell, Johnson, Lumens. ISBN: 1571762752

Appendix II

User's answers to the questions asked for my scenarios on Chapter 3, Page 13.

User A

Answer for Question 1)

Stability , Security and good scalability factors.

Also options to use easy graphical interface and to use it on another level similar to Slackware , where more control is possible.

Answer for Question 2)

1. Its windows. There for it tends to work in such a way that upgrades are necessary so Microsoft will constantly have a cash cow in there midst.

2. The structure tends to place every thing with the kernel , rather than taking a more layers approach to spread the risk in event of a system failure.

3.It is likely to be easily used as a zombie terminal in order to have its processing power as well as many others use to carry out cyber attacks. Windows being the most targeted o/s the computer market has ever known makes this disadvantage very possible.

4.Some security patches and updates risk to de-stabalise my current configuration due to the fact that such updates are being written for a very unstable and imperfect operating system to begin with. What developers base their development on how much money the product can make for them rather than how beneficial it can be for the spread of information technology.

5. Too many pointless processes running in the background for its own good.

Answer for Question 3)

The learning curve. Due to the fact that i am a windows user, i am more used to a simpler interface to the computer and therefore i am more prone to a simpler mentality when it comes to knowing what an o/s can be.

Although Linux in my opinion has by far a better philosophy to software development than Windows o/s it will take me time to think more like a Linux user rather than like a windows user.

Hardware compatibility is also a major issue.

Answer for Question 4)

I welcome the idea of a pre configured OS based on Linux. Because it would take me less time to get into it initially. Seeing as this approach would enable me to not start from scratch the time it would take me to get acquainted with such an o/s would be more favorable.

Answer for Question 5)

I would prefer it on the basis that it would give me more control in what i update. That way i can keep track of changes in my o/s configuration more easily.

User B

Answer for Question 1)

I expect my operating system to be compatible with everything I buy with the intention to use with that OS. Stability is always a concern, especially when I am doing something that matters outside of the computer system. I also expect that my OS has different modes such as a 'pretty-looking' mode that uses lots of resources and a 'sleek' mode where I can run just the bare bones to get by. I also expect it to work well with other machines, regardless of OS. Another thing I look for is the price and the security update process (eg. I don't want to have to wait for 2 months before an update comes out for a vulnerability.

Answer for Question 2)

The main disadvantage of my Current OS (Slackware Current) is that it can be very hard to learn, and you have to make a major commitment to learning the OS and how it works to get the full benefits. Also, it can be a pain to try and get certain things working (like ATI graphics cards) simply because many vendors are just starting to realize that people want to use Linux and they want it to work right, as opposed to the Windows perspective where "as long as it kind of works, i am fine". Other than that, i really don't see any disadvantages in the OS, it is more of the vendor's fault.

Answer for Question 4)

I think a Linux OS tailored specifically for my system would be amazing. To be honest, i think Linux could grow substantially if Dell and other computer manufacturers would include a cd of linux made for that computer-model. This would be amazing.

Answer for Question 5)

I don't. I use swaret (a Slackware package updating tool). On my windows box though, i hate having to look for updates manually, i think it is a waste of time and should, at the very least, be built into every program.

User C

Answer for Question 1)

I expect it to be a stable effective way to make use of my hardware with a variety of options to customise to my tastes for appearance

Answer for Question 2)

it's windows, there is no security, it's full of holes, 90 something percent of all virus's and other malware are written specifically for windows, there's constant patching which often doesn't work (some patches cause the system to be un-usuable) and others don't patch what it's supposed to, and some patched open more security holes

Answer for Question 3)

as far as changing to linux, compatability of hardware (drivers), knowing what software is available for what i need, adjusting to new ways to perform various tasks, and the new look and feel

Answer for Question 4)

I like the idea of a linux OS configured for my system (takes out all drivers issues and the like), it would be a good way to start to see what it's like and then move deeper into the config later

Answer for Question 5)

i DO NOT want something that will update for me, i do however like the idea of a nice clean straight forward way to find updates.... within in that tho it has to give access to everything that those updates include and give full control over which patches are used. Also, the patches should be kept so it is always possible to have a particular version i.e. it will not always force the latest update as the only possible

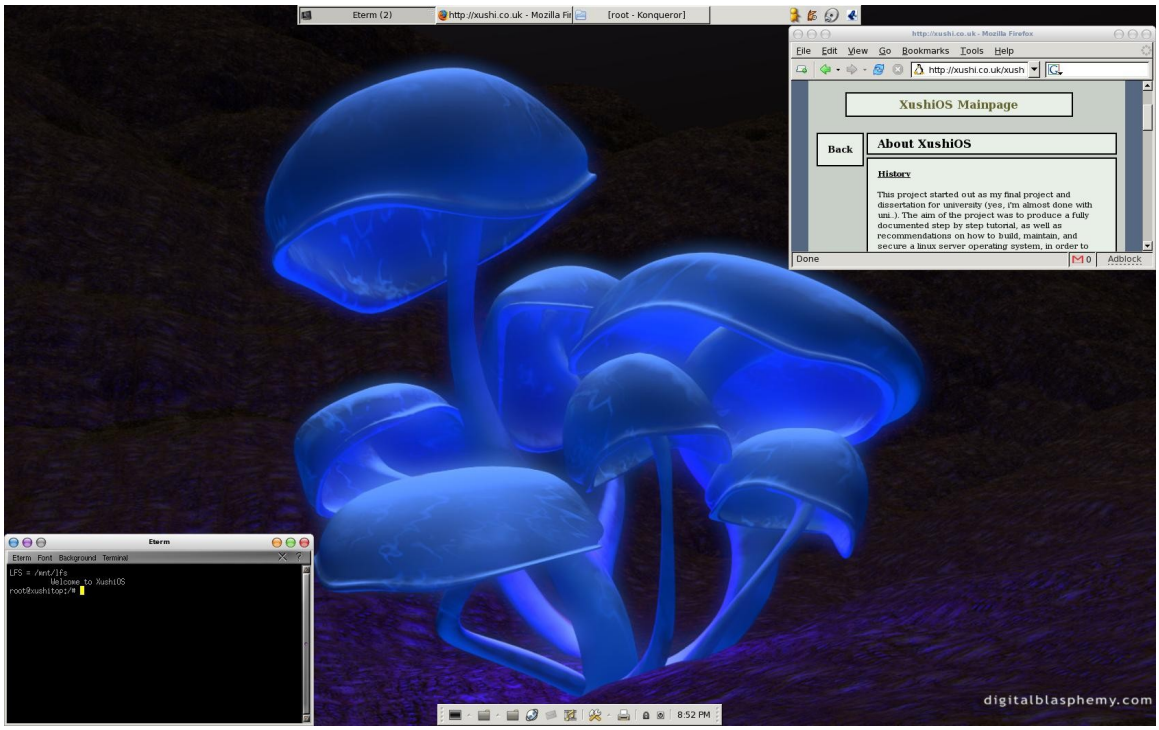
Appendix III
Images of Website for page 48

XushiOS main blog site, showing the latest information.

XushiOS website logo

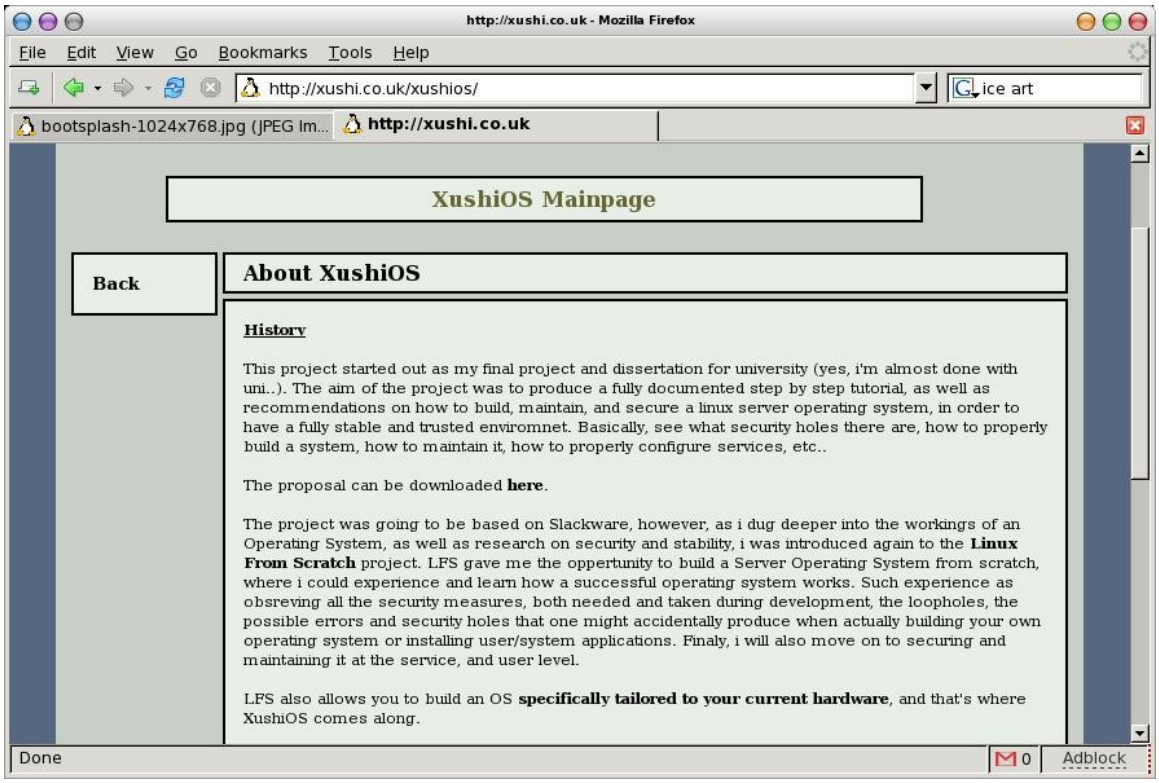
The Product:

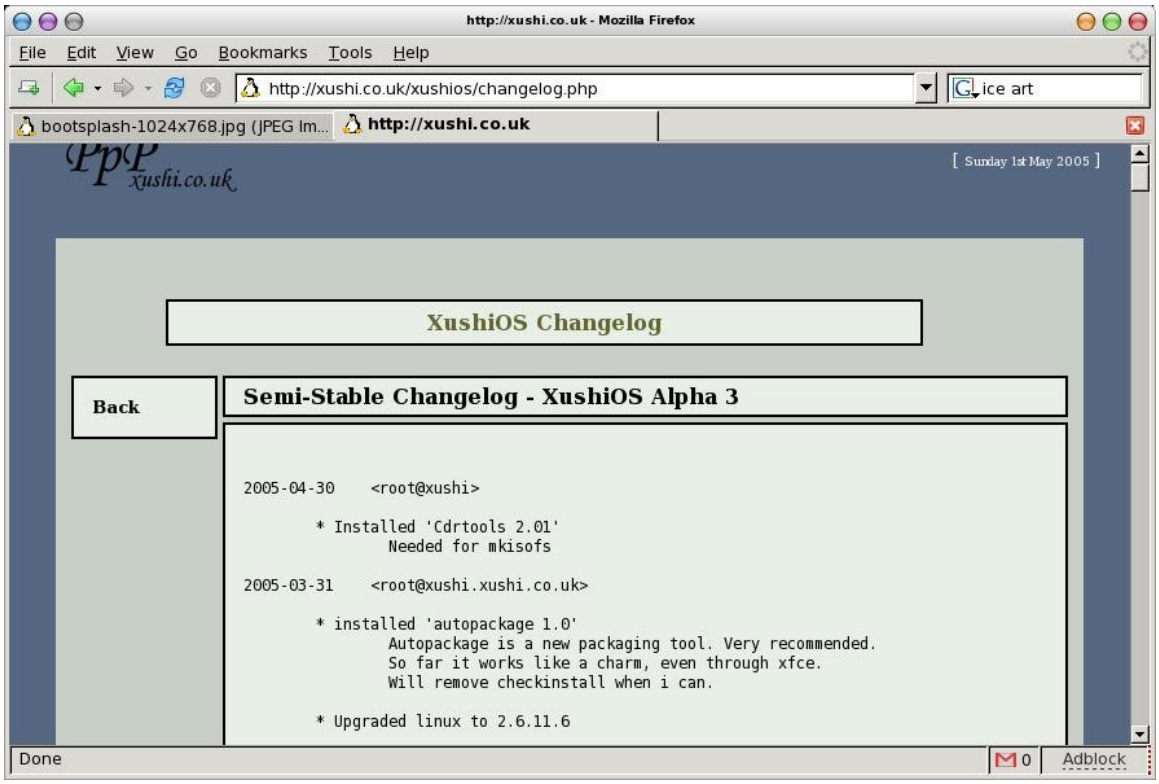
The user interface, with a shell and an Internet browser


Eterm (2)
[root - Konqueror]
File Edit View Go Bookmarks Tools Help
XushiOS Mainpage
Back
About XushiOS
History
This project started out as my final project and dissertation for university (yes, i'm almost done with uni..). The aim of the project was to produce a fully documented step by step tutorial, as well as recommendations on how to build, maintain, and secure a linux server operating system, in order to
Done
Adblock
Eterm
Eterm Font Background Terminal
LFS = /mnt/lfs
Welcome to XushiOS
8:52 PM
digitalblasphemy.com

The next page shows snapshots of XushiOS web page.

- XushiOS Main page
- XushiOS Changelog


http://xushi.co.uk - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://xushi.co.uk/xushios/
bootsplash-1024x768.jpg (JPEG Im...
http://xushi.co.uk
XushiOS Mainpage
Back
About XushiOS
History
This project started out as my final project and dissertation for university (yes, i'm almost done with uni..). The aim of the project was to produce a fully documented step by step tutorial, as well as recommendations on how to build, maintain, and secure a linux server operating system, in order to have a fully stable and trusted enviromnet. Basically, see what security holes there are, how to properly build a system, how to maintain it, how to properly configure services, etc..
The proposal can be downloaded here.
The project was going to be based on Slackware, however, as i dug deeper into the workings of an Operating System, as well as research on security and stability, i was introduced again to the Linux From Scratch project. LFS gave me the oppertunity to build a Server Operating System from scratch, where i could experience and learn how a successful operating system works. Such experience as obsreving all the security measures, both needed and taken during development, the loopholes, the possible errors and security holes that one might accidentally produce when actually building your own operating system or installing user/system applications. Finaly, i will also move on to securing and maintaining it at the service, and user level.
LFS also allows you to build an OS specifically tailored to your current hardware, and that's where XushiOS comes along.
Done
Adblock


http://xushi.co.uk - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://xushi.co.uk/xushios/changelog.php
bootsplash-1024x768.jpg (JPEG Im...
http://xushi.co.uk
xushi.co.uk
[Sunday 1st May 2005]
XushiOS Changelog
Back
Semi-Stable Changelog - XushiOS Alpha 3
2005-04-30 <root@xushi>
* Installed 'Cdrtools 2.01'
Needed for mkisofs
2005-03-31 <root@xushi.xushi.co.uk>
* installed 'autopackage 1.0'
Autopackage is a new packaging tool. Very recommended.
So far it works like a charm, even through xfce.
Will remove checkinstall when i can.
* Upgraded linux to 2.6.11.6
Done
Adblock

Thank you very much, and I hope you found my project interesting.

Samir Greadly

Word Count: 9,920 words
Created with OpenOffice.org Beta 2 (build 1.9.95)
A copy can be downloaded online from
http://xushi.co.uk/xushios/dissertation.pdf